Lecture Notes in Computer Science 12470

More information about this subseries at http://www.springer.com/series/7408

Bruno C. d. S. Oliveira (Ed.)

Programming Languages and Systems

18th Asian Symposium, APLAS 2020
Fukuoka, Japan, November 30 – December 2, 2020
Proceedings

Springer

Editor
Bruno C. d. S. Oliveira
University of Hong Kong
Hong Kong, Hong Kong

ISSN 0302-9743 ISSN 1611-3349 (electronic)
Lecture Notes in Computer Science
ISBN 978-3-030-64436-9 ISBN 978-3-030-64437-6 (eBook)
https://doi.org/10.1007/978-3-030-64437-6

LNCS Sublibrary: SL2 – Programming and Software Engineering

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Preface

This volume contains the papers presented at the 18th Asian Symposium on Programming Languages and Systems (APLAS 2020), held online during November 30 – December 2, 2020. APLAS 2020 was originally meant to be held in Fukuoka City, Japan, but due to the COVID-19 epidemic, it was changed to an online event.

APLAS aims to stimulate programming language research by providing a forum for the presentation of the latest results and the exchange of ideas in programming languages and systems. APLAS is based in Asia but is an international forum that serves the worldwide programming languages community.

This year we solicited contributions in the forms of regular research papers and tool papers. Among others, solicited topics include: semantics, logics, and foundational theory; design of languages, type systems, and foundational calculi; domain-specific languages; compilers, interpreters, and abstract machines; program derivation, synthesis, and transformation; program analysis, verification, model-checking; logic, constraint, probabilistic, and quantum programming; software security; concurrency and parallelism; tools and environments for programming and implementation; and applications of SAT/SMT to programming and implementation.

We also continued employing a light double-blind reviewing process adopted recently by APLAS with an author-response period. More precisely, we had a two-stage reviewing process. Each paper received at least three reviews before the author-response period, which was followed by a one-week Program Committee (PC) discussion, taking into account initial impressions of the papers as well as the author responses.

This year we received 46 submissions, out of which 19 papers (17 regular papers and 2 tool papers) were accepted after thorough reviews and discussions by the PC. We were also honored to include three invited talks by distinguished PL researchers:

- Luca Cardelli (University of Oxford, UK) on “Integrated Scientific Modeling and Lab Automation”
- Hidehiko Masuhara (Tokyo Institute of Technology, Japan) on “Object Support for GPU Programming: Why and How”
- Nadia Polikarpova (University of California San Diego, USA) on “Generating Programs from Types”

I am indebted to many people who helped make APLAS 2020 possible. First and foremost, I sincerely thank the PC, who have spent a lot of time and effort throughout the entire reviewing process. I am also grateful for the sub-reviewers and expert reviewers for their thorough and constructive reviews. I thank Masahiro Yasugi (Kyushu Institute of Technology, Japan) who served as a general chair and worked out every detail of the conference well in advance. This year’s APLAS was especially challenging to prepare due to the complications of moving to and organizing an online event.

I am also grateful to AAFS Executive Committee (especially Wei-Ngan Chin, National University of Singapore, Singapore, and Atsushi Igarashi, Kyoto University, Japan) who provided a lot of helpful advice and thank them for their leadership. I thank the previous APLAS PC chair, Anthony Widjaja Lin (TU Kaiserslautern, Germany) for his helpful advice and resources. Finally, I thank Eelco Visser and Elmer van Chastelet for their very helpful conf.researchr.org conference management system, as well as Eddie Kohler for his very helpful HotCRP conference management system.

October 2020 Bruno C. d. S. Oliveira

Organization

General Chair

Masahiro Yasugi	Kyushu Institute of Technology, Japan

General Vice-chair

Kento Emoto	Kyushu Institute of Technology, Japan

Local Arrangement Chair

Ryosuke Sato	The University of Tokyo, Japan

Remote Arrangement Chair

Tomoharu Ugawa	The University of Tokyo, Japan

Workshop Chair

Atsushi Igarashi	Kyoto University, Japan

Program Chair

Bruno C. d. S. Oliveira	The University of Hong Kong, Hong Kong

Program Committee

Edwin Brady	University of St Andrews, UK
Soham Chakraborty	IIT Delhi, India
Shigeru Chiba	The University of Tokyo, Japan
Andreea Costea	National University of Singapore, Singapore
Silvia Crafa	University of Padova, Italy
Pierre-Evariste Dagand	LIP6, CNRS, France
Mila Dalla Preda	University of Verona, Italy
Cristina David	University of Bristol, UK
Benjamin Delaware	Purdue University, USA
Jeremy Gibbons	University of Oxford, UK
Ichiro Hasuo	National Institute of Informatics, Japan
Sam Lindley	Heriot-Watt University and The University of Edinburgh, UK
James McKinna	The University of Edinburgh, UK
Madhavan Mukund	Chennai Mathematical Institute, India

Hakjoo Oh	Korea University, South Korea
Florian Rabe	University of Erlangen-Nuremberg, Germany
Sukyoung Ryu	KAIST, South Korea
Tom Schrijvers	KU Leuven, Belgium
Ilya Sergey	Yale-NUS College and National University of Singapore, Singapore
Marco Servetto	Victoria University of Wellington, New Zealand
Wouter Swierstra	Utrecht University, The Netherlands
Alwen Tiu	The Australian National University, Australia
Sam Tobin-Hochstadt	Indiana University Bloomington, USA
Janis Voigtländer	University of Duisburg-Essen, Germany
Meng Wang	University of Bristol, UK
Nicolas Wu	Imperial College London, UK
Yizhou Zhang	University of Waterloo, Canada
Tijs van der Storm	CWI, University of Groningen, The Netherlands

Additional Reviewer

Robert Rand

Abstracts of Invited Talks

Integrated Scientific Modeling and Lab Automation

Luca Cardelli

University of Oxford, UK
luca.a.cardelli@gmail.com

Abstract. The cycle of observation, hypothesis formulation, experimentation, and falsification that has driven scientific and technical progress is lately becoming automated in all its separate components. However, integration between these automated components is lacking. Theories are not placed in the same formal context as the (coded) protocols that are supposed to test them: neither description knows about the other, although they both aim to describe the same process. We develop integrated descriptions from which we can extract both the model of a phenomenon (for possibly automated mathematical analysis), and the steps carried out to test it (for automated execution by lab equipment). This is essential if we want to carry out automated model synthesis, falsification, and inference, by taking into account uncertainties in both the model structure and in the equipment tolerances that may jointly affect the results of experiments.

Object Support for GPU Programming: Why and How

Hidehiko Masuhara

Tokyo Institute of Technology, Japan
masuhara@is.titech.ac.jp

Abstract. General-purpose computing on graphics processing units (GPGPU) is now widely used in many application domains. However, programming for GPGPU is challenging due to its peculiar performance characteristics and still being done either in low-level languages or through libraries (e.g., those for matrix computation and machine learning). This talk discusses the performance challenges of using objects in GPGPU programming from the viewpoint of memory management, and the efficient mechanisms to support objects.

Generating Programs from Types

Nadia Polikarpova

UC San Diego, USA
npolikarpova@eng.ucsd.edu

Abstract. Program synthesis is a promising approach to automating low-level aspects of programming by generating code from high-level declarative specifications. But what form should these specifications take? In this talk I will advocate for using types as input to program synthesis. Types are widely adopted by programmers, they can vary in expressiveness and capture both functional and non-functional properties, and finally, type checking is often fully automatic and compositional, which helps the synthesizer find the right program. I will describe two type-driven program synthesizers we developed. The first one is Synquid, a synthesizer for recursive functional programs that uses expressive refinement types as a specification mechanism. The second one is Hoogle+, which relies on more mainstream Haskell types and generates code snippets by composing functions from Haskell libraries.

Contents

Program Generation, Transactions and Automation

Synthesis and Program Transformation

Debugging, Profiling and Constraint Solving